OIPE
AUG 31 2005
PATENT & TRADEMARK OFFICE

AB
2186
JFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:

Walter T. Nixon, et al.

Serial No. 10/027,353

Filed: December 19, 2001

For: CACHE ACCUMULATOR MEMORY WITH AN ASSOCIATIVITY MECHANISM

§ Group Art Unit: 2186
§
§ Examiner: Tsai, Sheng Jen
§
§ Atty. Dkt. No.: 5681-05300
§ P6846

CERTIFICATE OF MAILING
37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below:

Robert C. Kowert
Name of Registered Representative

August 29, 2005
Date

Signature

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal mailed June 27, 2005, Appellants present this Appeal Brief. Appellants respectfully request that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

As evidenced by the assignment recorded at Reel/Frame 012408/0178, the subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and now having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-36 are pending and finally rejected. The rejection of claims 1-36 is being appealed. A copy of claims 1-36 is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Computer and network systems may frequently perform operations on blocks of data ranging from tens to thousands of bytes or more. In some instances, such block operations may include accumulation operations in which an accumulator stores both an operand and a result of the operation. For example, in a RAID (Redundant Array of Inexpensive/Independent Disks) storage system, a block parity calculation may be implemented by performing an exclusive-OR (XOR) accumulation operation on all of the data words in a block. *See, e.g.*, specification, p. 1, line 9 – p. 2, line 9.

Because accumulation operations typically involve both a read and a write operation to an accumulator (e.g., to read the contents of the accumulator as an operand and to store a result within the accumulator), the accumulator may present a performance bottleneck for block accumulation operations. *See, e.g.*, specification, p. 2, lines 11-28. In conventional embodiments, an accumulator may be implemented using specialized components, such as high-bandwidth memory components (e.g., memory components having at least twice the access bandwidth as the desired throughput of block accumulation operations) or multi-ported memory components. However, such components may be expensive and may present their own efficiency challenges for certain patterns of read and write activity. *See, e.g.*, specification, p. 3, lines 1-29.

Independent claim 1 is directed to an apparatus comprising a memory, a functional unit configured to perform a block operation on one or more block operands to generate a block result, and a cache accumulator memory coupled to the memory and the functional unit, wherein the cache accumulator memory comprises a plurality of block storage locations, wherein the cache accumulator memory is configured to receive a set of one or more instructions to perform a first accumulation operation, wherein a first instruction in the set uses a first address in the memory to identify a first block operand. The cache accumulator memory is configured as a cache of the memory and is configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the first accumulation operation. In response to receiving the first instruction in the set, the cache accumulator memory is configured to access an associativity list comprising an indication that a first set of the block storage locations is allocated to the first accumulation operation and, in response to the indication, to provide the first block operand to the functional unit from the first set of block storage locations and to store the block result generated by the functional unit into the first set of block storage locations. One embodiment of such an apparatus is illustrated in FIG. 9 and described in the specification at p.31, line 14 – p. 33, line 5. In the illustrated embodiment, an apparatus is shown to include a cache accumulator memory 50A coupled to a memory 15 and to a functional unit 25 configured to perform a

block operation on one or more block operations to produce a block result. Cache accumulator memory 50A is configured as a cache of memory 15. With further reference to FIG. 11A and p. 34, line 24 – p. 35, line 9 of the specification, cache accumulator memory 50A is shown to include a number of addressable block storage locations and is configured to accumulate an intermediate result of a first accumulation operation, where the intermediate result is both a result of and an operand of the first accumulation operation. Cache accumulator memory 50A also includes an associativity list comprising a plurality of tags 45 respectively associated with the block storage locations. As described in the specification at p. 36, line 23 – p. 37, line 19, the associativity list may include an indication that a first set of the block storage locations is allocated to the first accumulation. In response to such an indication, cache accumulator memory 50A may provide a first block operand from the indicated block storage location to functional unit 25 and may store the block result generated by functional unit 25 to the indicated block storage location.

Independent claim 20 is directed to a method of performing a block accumulation operation using a cache accumulator memory that comprises a plurality of block storage locations, the method comprising receiving a first command in a set of commands used to implement an accumulation operation, wherein the first command is an instruction to perform an operation on a first block operand identified by a first address in a memory and to store a result of the operation, wherein the result is identified by a second address in the memory, and wherein the cache accumulator memory is configured as a cache of the memory; and, in response to said receiving a first command, accessing an associativity list comprising an indication that a first set of block storage locations of the cache accumulator memory is allocated to the first accumulation operation and, in response to the indication, providing the first block operand from the first set of block storage locations to a functional unit and storing a block result of the operation generated by the functional unit into the first set of block storage locations. The cache accumulator memory is configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the first

accumulation operation. One embodiment of such a method is illustrated in FIG. 17 and described in the specification at p. 42, line 20 – p. 43, line 23. Specifically, at block 1705, an instruction to perform an accumulation operation on a first block operand identified by a first address is received. At block 1707, an associativity list of a cache accumulator memory may be checked to determine whether the block operand is stored in the cache accumulator memory, and if so, the block operand may be provided from the cache accumulator memory to a functional unit and the block result generated by the functional unit may be stored in the cache accumulator memory, as shown in block 1715. As described above with respect to claim 1, the intermediate result of the block operation accumulated by the cache accumulator memory may be both a result of and an operand of the accumulation operation.

Independent claim 33 is directed to an apparatus comprising means for storing data, means for performing a block operation on one or more block operands to generate a block result, and means for storing the block result, wherein the means for storing the block result are coupled to the means for storing data and the means for performing a block operation, wherein the means for storing the block result comprise a plurality of block storage locations, wherein the means for storing the block result receive a first instruction comprised in a set of one or more instructions to perform a first accumulation operation, wherein the first instruction uses a first address in the means for storing data to identify a first block operand. The means for storing the block result is configured as a cache of the means for storing data, and is further configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the block accumulation operation. In response to the first instruction, the means for storing the block result access an associativity list that comprises an indication that a first set of the block storage locations is allocated to the first accumulation operation, wherein in response to the indication, the means for storing the block result provide the first block operand from the first set of block storage locations to the means for performing the block operation and store the block result in the first set of block storage locations. It is noted that the limitations of claim 33 generally

correspond to similar limitations of claim 1, and the portions of the specification and drawings referred to in support of claim 1 are also applicable to claim 33.

Independent claim 34 is directed to a data processing system comprising a host computer system, a storage array, an interconnect coupled to the host computer system and the storage array and configured to transfer data between the host computer system and the storage array, and a parity calculation system configured to perform parity operations on data stored to the storage array, wherein the parity calculation system comprises a memory, a cache accumulator memory, and a parity calculation unit. The cache accumulator memory comprises a plurality of block storage locations and is configured to receive a set of one or more instructions to perform a first accumulation operation, wherein a first instruction in the set uses a first address in the memory to identify a first block operand. The cache accumulator memory is configured as a cache of the memory and is further configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the first accumulation operation. In response to receiving the first instruction in the set, the cache accumulator memory is configured to access an associativity list comprising an indication that a first set of the block storage locations is allocated to the first accumulation operation and, in response to the indication, to provide the first block operand to the parity calculation unit from the first set of the block storage locations and to store the block result generated by the parity calculation unit into the first set of block storage locations. One embodiment of such a data processing system is illustrated in FIG. 1 and described in the specification at p. 11, line 3 – p. 13, line 8. As shown in FIG. 1, data processing system 300 includes a host 300 coupled via host/storage connection 304 to a storage system 306 including a storage array 308. Storage system 306 includes array controller 312 which, as shown in FIGs. 5 and 9, may include a memory 15, a cache accumulator memory 50/50A, and a functional unit 25 that may be configured to implement parity calculations. The remaining features of the cache accumulator memory recited in claim 34 are similar to those described above with respect to claim 1.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1, 4-6, 9-12, 16, 20, 23, 26, 27, 30, 33 and 34 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum et al. (U.S. Patent No. 4,888,679) (hereinafter "Fossum") in view of Sollars (U.S. Patent No. 6,216,218) (hereinafter "Sollars").

2. Claims 2, 19 and 21 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of McClure (U.S. Patent No. 5,590,307) (hereinafter "McClure").

3. Claims 3 and 22 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of Faraboschi et al. (U.S. Patent No. 6,122,708) (hereinafter "Faraboschi").

4. Claims 7, 8, 24 and 25 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of Handy, "The Cache Memory Book: The Authoritative Reference on Cache Design," Academic Press, 1993, page 57 (hereinafter "Handy").

5. Claims 13-15 and 28-29 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of "Microsoft Computer Dictionary," Microsoft Press, 2002, page 391: parity (hereinafter "Microsoft").

6. Claims 17, 18, 31 and 32 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of Morton (U.S. Patent No. 6,088,783).

7. Claims 35 and 36 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars and Morton, and further in view of Microsoft.

VII. ARGUMENT

First Ground of Rejection:

Claims 1, 4-6, 9-12, 16, 20, 23, 26, 27, 30, 33 and 34 are rejected under 35 U.S.C. § 103(a) Fossum in view of Sollars. Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 4, 16, 20, 23, 30 and 33:

In regard to claim 1, contrary to the Examiner's assertion, the cited art fails to teach or suggest a cache accumulator memory coupled to a memory and a functional unit, wherein the cache accumulator memory comprises a plurality of block storage locations, wherein the cache accumulator memory is configured to receive a set of one or more instructions to perform a first accumulation operation, wherein a first instruction in the set uses a first address in the memory to identify a first block operand; wherein the cache accumulator memory is configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the first accumulation operation; and wherein in response to receiving the first instruction in the set, the cache accumulator memory is configured to access an associativity list comprising an indication that a first set of the block storage locations is allocated to the first accumulation operation and, in response to the indication, to provide the first block operand to the functional unit from the first set of block storage locations and to store the block result generated by the functional unit into the first set of block storage locations.

Neither Fossum nor Sollars teach or suggest a cache accumulator memory configured to accumulate an intermediate result of an accumulation operation that is both a result of and an operand of the accumulation operation. In the Final Action mailed March 28, 2005, the Examiner acknowledges that Fossum does not disclose that a cache memory is configured to accumulate an intermediate result that is both a result and

an operand, but argues that such behavior is "exactly the intended purpose of a cache memory," in that caches "store data that is needed currently, or soon to be needed... [and] thus intermediate results will certainly be accumulated and stored in the cache memory." Appellants strongly disagree with the Examiner's interpretation of Fossum and the Examiner's reasoning in view of the specific language of Appellants' claim. That a cache may be configured to store data that is produced by one operation and utilized by another operation is not in any way suggestive of a cache accumulator memory that is configured to accumulate an intermediate result of an accumulation operation, where the intermediate result is both a result of and an operand of the accumulation operation. The Examiner seems to suggest that storing data in a cache is equivalent to an accumulation operation. However, Appellants have recited a specific relationship between the result of an accumulation operation and an operand of the accumulation operation that is not in any way inherent in the data storage operation of a cache as a cache is conventionally understood or disclosed by Fossum.

Fossum is directed specifically to the problem of data movement between a main memory and a cache in a vector processor (Fossum, Abstract). With respect to vector operations, Fossum discloses only a vector processing unit 116 including as functional subunits only a vector adder 119, a vector mask unit 120 and a vector multiplier 121 (Fossum, FIG. 7 and col. 12, lines 1-30). Appellants note that Fossum omits any discussion of the notion of vector or block *accumulation*. As described above and in the background section of Appellants' specification, block accumulation operations have unique data characteristics that present specific implementation challenges distinct from other types of block operations, such as addition and multiplication operations. Fossum fails to acknowledge any aspect of these challenges, and is directed to an entirely separate problem. As such, not only does Fossum fail to teach or suggest the limitations of claim 1, he provides no motivation or support in favor of a solution to the problem addressed by Appellants' claim 1.

The Examiner cites Sollars as independent evidence that a cache memory "is intended for storing intermediate results and for supplying operands for next operation [sic]." However, Appellants' claim specifically recites that the accumulated intermediate result is not merely an operand of an arbitrary "next operation," but is in fact both a result of and an operand of a single block accumulation operation. Sollars makes no mention of the problem of block operations in general or block accumulation operations in particular, and as argued above, a cache memory is not suggestive, in and of itself, of a cache accumulator memory for block accumulation operations.

In an Advisory Action mailed June 10, 2005, the Examiner argues that an "...accumulation operation is well known in the art and widely used to calculate the sum of a sequence of numbers, such as a vector, which is the specific domain of Fossum et al's invention." As argued above, vector addition as mentioned by Fossum is not equivalent to accumulation, and Fossum makes no specific mention of accumulation (as admitted previously by the Examiner). Further, Appellants' claim does not focus solely on an accumulation operation in isolation, but rather recites a specific structure – a cache accumulator memory – that includes a combination of caching and accumulation features with respect to block accumulation operations that are simply not taught or suggested by the combination of the cited art.

Additionally, neither Fossum nor Sollars teaches or suggests that a cache accumulator memory is configured to access an associativity list comprising an indication that a first set of the block storage locations is allocated to a first accumulation operation, and that the cache accumulator memory is further configured to responsively provide a first block operand to the functional unit and to store the block result generated by the functional unit into the first set of block storage locations. The Examiner relies upon Fossum, col. 4, lines 25-47 to teach this limitation. However, the cited portion of Fossum merely describes the general behavior of a cache in determining whether a data block is resident within the cache, and if not,

fetching the data block from a main memory for storage within the cache. This is in no way suggestive of the associativity list recited in Appellants' claim.

Fossum describes the structure of the cache in further detail in col. 6, lines 20-63 and FIG. 2. However, Fossum discloses only that a cache may include a tag store 32 in conjunction with a data store 33, wherein the tag store 32 stores address bits ("tag bits") associated with data blocks stored within the data store 33, which tag bits are used as the basis for determining whether a given data block is resident within the cache. Fossum's tag bits are in no way suggestive of an associativity list that includes an indication that a set of block storage locations is allocated to a particular accumulation operation. As argued above, Fossum does not teach or suggest any aspect of block accumulation operations. Further, Fossum's tag bits are not indicative of any type of association between a data block and an operation. Instead, they merely identify the data block itself for the purpose of determining a block's presence or absence within the cache.

Sollars also fails to teach or suggest an associativity list as recited in Appellants' claim. Thus, for at least the foregoing reasons, Appellants submit that claim 1 is patentably distinguishable over the cited art. A similar argument also applies to independent claims 20 and 33.

Claims 5, 6 and 9:

In addition to the reasons given above for claim 1, the cited art fails to teach or suggest the limitation recited in claim 5 in which the cache accumulator memory is configured to load a copy of the first block operand into the first set of block storage locations in the cache accumulator memory from the memory in response to the first block operand not being present in the cache accumulator memory when the first instruction is received. The Examiner relies on Fossum to teach this limitation. However, Fossum is directed to a prefetch mechanism in which a data vector is prefetched from memory in response to a vector load instruction, not in response to whether the data is currently located in the cache. Specifically, Fossum notes that

> [i]n general, the required blocks are requested before the corresponding vector elements are addressed by the vector processor, and in some cases the blocks will have been fetched from memory by the time that the vector processor 22 addresses the vector elements specified by the vector load instruction. (col. 5, lines 37-42, emphasis added)

That is, Fossum does not condition a block operand load on whether that operand is present in a cache accumulator memory, and by unconditionally prefetching in response to the vector load instruction, in fact appears to assume that the block operand is never present in its entirety within a cache. Thus, for at least the foregoing reasons, Appellants submit that the rejection of claims 5, 6 and 9 is clearly unsupported by the cited art.

Claims 10 and 26:

In addition to the reasons given above for claim 1, Fossum in view of Sollars fails to teach or suggest all of the limitations of claim 10. Specifically, claim 10 recites that the cache accumulator memory updates the associativity list in response to storing the block result generated by the functional unit. In rejecting claim 10, the Examiner simply equates the distinct limitations recited in claims 9 and 10 with respect to loading into the cache accumulator memory from the memory versus updating the associativity list in response to storing a block result within the cache accumulator memory, and asserts that the teachings of Fossum are applicable to both cases. However, Fossum provides no discussion whatsoever regarding the details of storing block results from a functional unit within a cache accumulator memory or any other type of memory. As noted above, Fossum is directed to the problem of prefetching data inputs for a vector processor, and is silent with respect to the problem of managing the output of such a processor. Fossum clearly provides no support for the specific recitations of claim 10, and the Examiner's assertion regarding the behavior of Fossum's processor in this situation is pure speculation in hindsight. A similar argument applies to claim 26 having limitations similar to claim 10. Therefore, Appellants submit that the rejection of claims 10 and 26 is not supported by the cited art.

Claims 11, 12 and 27:

In addition to the reasons given above for claim 10, Fossum in view of Sollars fails to teach or suggest all of the limitations of claim 11. Claim 11 recites that updating the associativity list in response to storing a block result includes updating a tag by setting the tag equal to a first portion of address bits of an address in memory that identifies the block result. Although Fossum describes a tag field and its behavior with respect to a vector load operation, Fossum provides no support whatsoever for the recitations of claim 11 with respect to a block result generated by a functional unit. As noted above with respect to claim 10, Fossum is silent regarding the case of storage of block results. In rejecting claim 11, the Examiner cites an example regarding an ADD instruction. However, the Examiner's example is not at all grounded in the specific disclosure of Fossum. As with claim 10 above, the Examiner appears to be speculating, in hindsight, as to what Fossum might do in an area in which Fossum fails to provide any substantive support. A similar argument applies to claim 27 having limitations similar to claim 11. Thus, Appellants submit that the rejection of claims 11, 12 and 27 is not supported by the cited art.

Claim 34:

Appellants note that claim 34 recites features similar to those of claim 1 within a context that includes other elements, include a host computer system, a storage array, an interconnect coupled to the host computer system and storage array, and a parity calculation system configured to perform parity operations on data stored to the storage array. However, the Examiner has not attempted to demonstrate that the combination of Fossum and Sollars discloses any of these additional features, and in fact the cited references fail to do so either separately or in combination. Appellants submit that no *prima facie* rejection has ever been stated for claim 34 since the Examiner never addressed the additional features of claim 34 in combination. Moreover, Appellants

submit that the rejection of claim 34 is not supported by the cited art for at least these reasons in addition to the reasons given above with respect to claim 1.

Second Ground of Rejection:

Claims 2, 19 and 21 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of McClure. Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 2 and 19:

Appellants submit that the rejection of claims 2 and 19 is not supported by the cited art for at least the reasons given above with respect to claim 1.

Claim 21:

In addition to the reasons given above for claim 20, the cited art fails to teach or suggest all of the limitations recited in claim 21. In particular, claim 21 recites that the act of storing a block result into a cache accumulator memory comprises overwriting a first block operand with the block result. Although the Examiner rejected claim 21 on the same grounds as claims 2 and 19, the cited references contain no support for this particular limitation that does not appear in either claim 2 or claim 19. Therefore, the rejection of claim 21 is not supported by the cited art. Moreover, the Examiner has never even stated a *prima facie* rejection of claim 21 since the Examiner never acknowledged the difference in scope between claim 21 and claims 2 and 19.

Third Ground of Rejection:

Claims 3 and 22 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of Faraboschi. Appellants traverse this rejection for the following reasons.

In addition to the reasons given above for claim 1, the cited references fail to teach or suggest the limitations recited in claim 3. Specifically, the cited references fail to disclose a cache accumulator memory comprising at least two independently interfaced memory banks, wherein the cache accumulator memory is configured to provide a block operand from a first one of the memory banks and to store the block result in a second one of the memory banks. In rejecting claim 3, the Examiner refers to block interleaving of memory bank addresses for read operations as discussed in Fossum, and asserts that Faraboschi discloses the claimed limitation. However, block interleaving of addresses has for read operations as discussed in Fossum has nothing to do with the features recited in claim 3, which specifically state that a block operand is read from a first memory bank and a block result is written to a second memory bank. Faraboschi discloses only that a data cache may include a number of banks that may be separately decoded, and does not disclose in any way the specific relationship of reading operands and writing results to those separate banks as recited in claim 3. A similar argument applies to claim 22. Appellants therefore submit that the rejection of claims 3 and 22 is not supported by the cited art.

Fourth Ground of Rejection:

Claims 7, 8, 24 and 25 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of Handy. Appellants traverse this rejection and submit that the rejection of these claims is not supported by the cited art for at least the reasons given above for claims 1 and 20.

Fifth Ground of Rejection:

Claims 13-15 and 28-29 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of Microsoft. Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 13 and 28:

Appellant submits that the rejection of claims 13 and 28 is not supported by the cited art for at least the reasons given above with respect to claims 1 and 20.

Claims 14, 15 and 29:

In addition to the reasons given above for claim 1, the cited references fail to teach or suggest the limitation of claim 14 in which the command is issued by a storage system controller. In rejecting claim 14, the Examiner asserts that instructions/commands are provided in Fossum by an instruction processing unit 107. However, an instruction processing unit of a microprocessor is in no way suggestive of a storage system controller that issues commands or operations to a cache accumulator memory. A similar argument applies to claim 29. Therefore, Appellants submit that the rejection of claims 14, 14 and 296 is not supported by the cited art.

Sixth Ground of Rejection:

Claims 17, 18, 31 and 32 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars, and further in view of Morton. Appellants traverse this rejection and submit that the rejection of these claims is not supported by the cited art for at least the reasons given above with respect to claims 1 and 20.

Seventh Ground of Rejection:

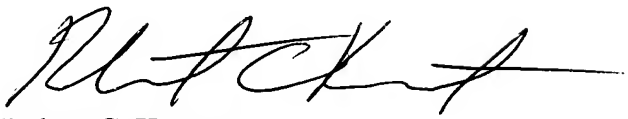
Claims 35 and 36 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Fossum in view of Sollars and Morton, and further in view of Microsoft. Appellants traverse this rejection and submit that the rejection of these claims is not supported by the cited art for at least the reasons given above with respect to claim 34.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-36 was erroneous, and reversal of the Examiner's decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-05300/RCK. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850
Date: August 29, 2005

IX. CLAIMS APPENDIX

The claims on appeal are as follows.

1. An apparatus, comprising:

a memory;

a functional unit configured to perform a block operation on one or more block operands to generate a block result; and

a cache accumulator memory coupled to the memory and the functional unit, wherein the cache accumulator memory comprises a plurality of block storage locations, wherein the cache accumulator memory is configured to receive a set of one or more instructions to perform a first accumulation operation, wherein a first instruction in the set uses a first address in the memory to identify a first block operand;

wherein the cache accumulator memory is configured as a cache of the memory;

wherein the cache accumulator memory is configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the first accumulation operation; and

wherein in response to receiving the first instruction in the set, the cache accumulator memory is configured to access an associativity list comprising an indication that a first set of the block storage locations is allocated to the first accumulation operation and, in response to the indication, to provide the first block operand to the functional unit from

the first set of block storage locations and to store the block result generated by the functional unit into the first set of block storage locations.

2. The apparatus of claim 1, wherein the cache accumulator memory comprises a dual-ported memory.

3. The apparatus of claim 1, wherein the cache accumulator memory comprises at least two independently interfaced memory banks, wherein the cache accumulator memory is configured to provide the first block operand from a first block storage location in a first one of the independently interfaced memory banks and to store the block result in a second block storage location in a second one of the independently interfaced memory banks, wherein the first set of block storage locations comprises the first block storage location and the second block storage location.

4. The apparatus of claim 1, wherein the cache accumulator memory is configured to indicate whether a particular block operand stored in the cache accumulator memory is modified with respect to a copy of that particular block operand in the memory.

5. The apparatus of claim 1, wherein the cache accumulator memory is configured to load a copy of the first block operand into the first set of block storage locations in the cache accumulator memory from the memory in response to the first block operand not being present in the cache accumulator memory when the first instruction is received.

6. The apparatus of claim 5, wherein if all of the block storage locations in the cache accumulator memory are currently storing valid data when the first instruction is received, the cache accumulator memory is configured to select the first set of block storage locations and to load the copy of the first block operand into the first set of block storage locations, wherein the cache accumulator memory is further configured to update

the indication in the associativity list to indicate that the first set of the block storage locations is allocated to the first accumulation operation in response to selecting the first set of block storage locations.

7. The apparatus of claim 6, wherein the cache accumulator memory is configured to use a least recently used algorithm to select the first set of block storage locations to overwrite.

8. The apparatus of claim 6, wherein if data to be overwritten in the first set of block storage locations is modified with respect to a copy of that data in the memory, the cache accumulator memory is configured to write the data back to the memory before loading the copy of the first block operand into the first set of block storage locations.

9. The apparatus of claim 5, wherein in response to loading the first block operand into the first set of block storage locations, the cache accumulator memory is configured to update a tag associated with the first set of block storage locations to indicate that the first block operand is stored within.

10. The apparatus of claim 1, wherein the cache accumulator memory is configured to update the associativity list in response to storing the block result generated by the functional unit, wherein the cache accumulator memory is configured to update the associativity list by updating a tag associated with the first set of block storage locations to indicate that the block result is stored within the first set of block storage locations.

11. The apparatus of claim 10, wherein the cache accumulator memory is configured to update the tag by setting the tag to equal a first portion of address bits of a second address in the memory that identifies the block result.

12. The apparatus of claim 11, wherein the second address is not equal to the first address.

13. The apparatus of claim 1, wherein the functional unit is configured to perform a parity calculation on the block operand.

14. The apparatus of claim 1, wherein the operation comprises a parity calculation, and wherein the command is issued by a storage system controller.

15. The apparatus of claim 14, wherein the functional unit is configured to calculate a parity block from a plurality of data blocks in a stripe of data, wherein the first block operand is a first one of the data blocks in the stripe of data.

16. The apparatus of claim 1, wherein the functional unit is configured to perform the operation on two block-operands.

17. The apparatus of claim 16, wherein a first of the two block-operands is the first block operand stored in the cache accumulator memory and a second of the two block-operands is provided on a data bus coupled to provide operands to the functional unit.

18. The apparatus of claim 16, wherein a first of the two block-operands is the first block operand stored in the cache accumulator memory and a second of the two block-operands is provided from the memory.

19. The apparatus of claim 1, wherein the cache accumulator memory is configured to store a word of the block result during an access cycle in which cache accumulator memory also provides a word of the first block operand to the means for performing a block operation.

20. A method of performing a block accumulation operation using a cache accumulator memory that comprises a plurality of block storage locations, the method comprising:

receiving a first command in a set of commands used to implement an accumulation operation, wherein the first command is an instruction to perform an operation on a first block operand identified by a first address in a memory and to store a result of the operation, wherein the result is identified by a second address in the memory, and wherein the cache accumulator memory is configured as a cache of the memory;

in response to said receiving a first command:

accessing an associativity list comprising an indication that a first set of block storage locations of the cache accumulator memory is allocated to the first accumulation operation;

in response to the indication, providing the first block operand from the first set of block storage locations to a functional unit and storing a block result of the operation generated by the functional unit into the first set of block storage locations;

wherein the cache accumulator memory is configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the first accumulation operation.

21. The method of claim 20, wherein the cache accumulator memory comprises a dual-ported memory, wherein said storing comprises overwriting the first block operand with the block result.

22. The method of claim 20, wherein the cache accumulator memory comprises at least two independently interfaced memory banks, wherein said loading comprises loading the first block operand into a first block storage location in a first one of the independently interfaced memory banks and wherein said storing comprises storing the block result in a second block storage location in a second one of the independently interfaced memory banks, wherein the first set of block storage locations comprises the first block storage location and the second block storage location.

23. The method of claim 20, further comprising selecting the first set of block storage locations and loading the first block operand into the first set of block storage locations if all of the block storage locations are currently storing valid data when the first command is received.

24. The method of claim 23, wherein said selecting comprises using a least recently used algorithm to select the first set of block storage locations.

25. The method of claim 23, further comprising writing data in the first set of block storage locations back to the memory if the data is modified with respect to a copy of that data in the memory.

26. The method of claim 20, further comprising updating the indication in the associativity list by updating a tag associated with the first set of block storage locations to indicate that the block result is stored within the first set of block storage locations in response to storing the block result generated by the functional unit.

27. The method of claim 26, wherein said updating the tag comprises setting the tag to equal a first portion of address bits of the second address.

28. The method of claim 20, further comprising the functional unit performing a parity calculation on the first block operand to generate the block result in response to said providing.

29. The method of claim 20, wherein the operation comprises a parity calculation, and wherein the command is issued by a storage system controller.

30. The method of claim 20, further comprising the functional unit performing the operation on the first block operand and a second block operand in response to said providing.

31. The method of claim 30, further comprising a data bus providing the second of the two block operands to the functional unit.

32. The method of claim 30, further comprising the memory providing the second of the two block operands to the functional unit.

33. An apparatus, comprising:

means for storing data;

means for performing a block operation on one or more block operands to generate a block result; and

means for storing the block result, wherein the means for storing the block result are coupled to the means for storing data and the means for performing a block operation, wherein the means for storing the block result comprise a plurality of block storage locations, wherein the means for storing the block result receive a first instruction comprised in a set of one or more instructions to perform a first accumulation operation, wherein the first

instruction uses a first address in the means for storing data to identify a first block operand;

wherein the means for storing the block result is configured as a cache of the means for storing data;

wherein the means for storing the block result is configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the block accumulation operation; and

wherein in response to the first instruction, the means for storing the block result access an associativity list that comprises an indication that a first set of the block storage locations is allocated to the first accumulation operation, wherein in response to the indication, the means for storing the block result provide the first block operand from the first set of block storage locations to the means for performing the block operation and store the block result in the first set of block storage locations.

34. A data processing system, comprising:

a host computer system;

a storage array;

an interconnect coupled to the host computer system and the storage array and configured to transfer data between the host computer system and the storage array; and

a parity calculation system configured to perform parity operations on data stored to the storage array, wherein the parity calculation system comprises a memory, a cache accumulator memory, and a parity calculation unit;

wherein the cache accumulator memory comprises a plurality of block storage locations and is configured to receive a set of one or more instructions to perform a first accumulation operation, wherein a first instruction in the set uses a first address in the memory to identify a first block operand;

wherein the cache accumulator memory is configured as a cache of the memory;

wherein the cache accumulator memory is configured to accumulate an intermediate result of the first accumulation operation, wherein the intermediate result is both a result of and an operand of the first accumulation operation; and

wherein in response to receiving the first instruction in the set, the cache accumulator memory is configured to access an associativity list comprising an indication that a first set of the block storage locations is allocated to the first accumulation operation and, in response to the indication, to provide the first block operand to the parity calculation unit from the first set of the block storage locations and to store the block result generated by the parity calculation unit into the first set of block storage locations.

35. The data processing system of claim 34, wherein the parity calculation unit is configured to perform a parity calculation on the first block operand provided by the cache accumulator memory and a second block operand provided on a data bus.

36. The data processing system of claim 31, wherein the parity calculation system is configured to calculate a parity block from a plurality of data blocks in a stripe of data when performing the first accumulation operation, wherein the first block operand is a first one of the data blocks in the stripe of data and wherein the second block operand is a second one of the data blocks in the stripe of data.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.